UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/874,170	06/04/2001	Vasanth Bala	10003355-1	7644

7590 10/10/2007

HEWLETT-PACKARD COMPANY
Intellectual Property Administration
P.O. Box 272400
Fort Collins, CO 80527-2400

EXAMINER	
PROCTOR, JASON SCOTT	
ART UNIT	PAPER NUMBER
2123	
MAIL DATE	DELIVERY MODE
10/10/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	09/874,170	BALA ET AL.	
	Examiner	Art Unit	
	Jason Proctor	2123	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *25 June 2007*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *2-24* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *2-24* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *04 June 2001* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

Claims 2-24 were rejected in the Office Action entered on 30 January 2007.

The response submitted on 7 May 2007 included amendments and remarks. The amendments were not entered for being noncompliant with 37 CFR 1.121 because of an incorrect status identifier for claim 18. In response, Applicants submitted a corrected listing of the claims on 25 June 2007.

This Office Action is in response to the 7 May 2007 remarks and the 25 June 2007 listing of the claims.

The 25 June 2007 submission has amended claim 23.

Claims 2-24 are pending in this application.

Claims 2-24 are rejected.

Claim Interpretation

1. It is not entirely clear how to identify whether a code segment has been "dynamically tailored" as recited by at least claims 2, 14, and 18. It is unclear if this distinguishes the granularity from other types of tailoring or other dynamic processes. Claim 1 recites "parsing of said code segments being dynamically performed" and later recites "the plurality of dynamically tailored code segments," which appears to imply that "tailor" means "parse" in the context of the claim language. It is unclear whether Applicants intend for these terms to be interchangeable or if one term should be interpreted more broadly than the other. For the purposes of examination,

the phrase "dynamically tailored" is interpreted as meaning, "parsed upon request, rather than in advance of a request" and generally conforms to the nature of the invention (abstract).

Response to Arguments – 35 USC § 103

2. In response to the previous rejection of claims 2-24 under 35 U.S.C. § 103 as being unpatentable over US Patent No. 6,370,687 to Shimura in view of Official Notice, Applicants argue primarily that:

> Applicants respectfully submit that Claim 2 (Claims 9, 14, 18, and 23 include similar features) includes the features "a server code segment manager coupled to the application code transformation manager, for parsing the client application in the native binary format into a plurality of code segments, said parsing of said code segments being dynamically performed based on actual server-side and client-side execution overhead, network bandwidth efficiency and client-side storage requirements on a per client basis, and configured based on predicted code segment usage or prior code segment usage history (emphasis added)." [...]
>
> Applicants respectfully submit that Shimura does not teach this claimed feature. Applicants do not understand Shimura to teach dynamic parsing of application code into code segments wherein the parsing of the code segments is dynamically performed based on actual server-side and client-side execution overhead, network bandwidth efficiency and client-side storage requirements on a per client basis (emphasis added).

The Examiner respectfully traverses this argument as follows.

Shimura plainly teaches dynamically compiling the application code, that is, compiling the code based on a dynamic request for the application (column 2, lines 10-29). Shimura plainly teaches that said request come from a client, therefore the dynamic compiling is performed on a per client basis (column 4, lines 11-19). Therefore what is left to determine is whether Shimura teaches that dynamic parsing of code segments, performed on a per client basis, is performed based on actual server-side and client-side execution overhead, network bandwidth efficiency and client-side storage requirements (emphasis added). The breadth of this limitation

hinges upon the phrase "based on" which is clearly expansive. The Shimura reference teaches each of these claimed limitations for the following reasons.

The Shimura reference teaches performing dynamic parsing of code segments ***based on actual server-side and client-side execution overhead*** because parsing and transmitting (by the server) and receiving and executing (by the client) are all instances of actual server-side and client-side execution overhead. Alternatively, if the server has no available execution overhead because it is operating at its capacity, it would be impossible for the server to also dynamically parse code segments. Similarly, if the client has no available execution overhead because it is operating at its capacity, it would be impossible for the client to receive and execute the code segments. Therefore, in the system taught by Shimura, the server performs dynamic parsing of code segments on a per client basis ***based on actual server-side and client-side execution overhead***, as claimed.

The Shimura reference teaches performing dynamic parsing of code segments ***based on actual network bandwidth efficiency*** because parsing and transmitting (by the server) and receiving and executing (by the client) requires available network bandwidth. Alternatively, if the network is inefficient and has no available bandwidth, it would be impossible for the client to request and for the server to transmit a parsed code segment. Therefore, in the system taught by Shimura, the server performs dynamic parsing of code segments on a per client basis ***based on actual network bandwidth efficiency***, as claimed.

The Shimura reference teaches performing dynamic parsing of code segments ***based on actual client-side storage requirements*** because the client receives and executes parsed code segments. Alternatively, if the client has no available storage capacity, it would be impossible

for the client to receive and execute a parsed code segment. Therefore, in the system taught by Shimura, the server performs dynamic parsing of code segments on a per client basis ***based on actual client-side storage requirements***, as claimed.

Amendments to the claim language which limit what is meant by "based on" in the present claim language may overcome the Shimura reference. However, the current claim language makes no explicit requirement for the dynamic parsing of code segments with respect to execution overhead, network efficiency, or storage requirements that are not taught by Shimura. The Shimura reference teaches the limitations referenced in Applicants' arguments.

Regarding claim 6, Applicants argue primarily that:

> Applicants readily agree with the Examiner that the obvious details of implementation are omitted by the teachings of Shimura. Applicants submit that the obvious omission of the details of implementation of the features of Claim 6 clearly illustrate that the Claimed features were not taught or rendered obvious over Shimura.

The Examiner respectfully traverses this argument as follows.

There appears to be agreement that the limitations of claim 6 are regarded as the ***obvious*** details of implementing the invention of claim 2, from which claim 6 depends. The Examiner submits that, in accordance with Graham v. Deere and 35 U.S.C. § 103, this agreement clearly suggests that claim 6 is not patentable over Shimura and Official Notice used in the rejection of claim 2.

In claim 6, Applicants have chosen to describe features that, although not explicitly described by Shimura, would have been obvious to a person of ordinary skill in the art. Claim 6 is therefore unpatentable over Shimura.

As an example of what is known by artisans of ordinary skill and not relied upon in the rejection under 35 U.S.C. § 103, "Operating System Concepts, Fifth Edition" by Abraham Silberschatz and Peter Baer Galvin, 1999, is an ubiquitous undergraduate text commonly referred to as "The Dinosaur Book" (for its colorful front cover) among those skilled in the art of computer programming. Silberschatz describes memory management schemes in chapter 8, and in particular "Dynamic Loading" and "Dynamic Linking" (pages 242-243, sections 8.1.2 and 8.1.3). Silberschatz describes:

> [...] With dynamic loading, a routine is not loaded until it is called... When a routine needs to call another routine, the calling routine first checks to see whether the other routine has been loaded. If it has not been, the relocatable linking loader is called to load the desired routine into memory and to update the program's address tables to reflect this change. Then, control is passed to the newly loaded routine.
> [...] This scheme is particularly useful when large amounts of code are needed to handle infrequently occurring cases, such as error routines. In this case, although the total program size may be large, the portion that is actually used (and hence actually loaded) may be much smaller.
> [...] Dynamic loading does not require special support from the operating system.
> [...] The concept of dynamic linking is similar to that of dynamic loading. Rather than loading being postponed until execution time, linking is postponed. [...] With dynamic linking, a *stub* is included in the image for each library-routine reference. This stub is a small piece of code that indicates how to locate the appropriate memory-resident library routine, or how to load the library if the routine is not already present.
> [...] When this stub is executed, it checks to see whether the needed routine is already in memory. If the routine is not in memory, the program loads it into memory. Either way, the stub replaces itself with the

> address of the routine, and executes the routine. Thus, the next time that that code segment is reached, the library routine is executed directly, incurring no cost for dynamic linking.

Because these teachings are found in a ubiquitous undergraduate level text and teach the limitations of claim 6, these teachings are well within the knowledge of a person of ordinary skill in the art of computer programming.

Applicants traverse the Examiner's use of Official Notice regarding claim 7 by further argue that:

> Applicants respectfully submit that the claimed embodiments of "adjusting any branch targets in code segments stored in the code cache that need to branch to the received code segment and had previously been adjusted to branch out of the code cache to the client code segment manager to now branch to appropriate locations within the received code segment; adjusting any branch instructions in the received code segment having branch targets that branch to code segments currently in the code cache to branch to the appropriate code segments in the code cache; and adjusting any branch instructions in the received code segment having branch targets that need to branch to code segments not in the code cache to branch out of the code cache to the client code segment manager to request the code segment containing the branch targets" is not considered to be common knowledge or well-known in the art, as asserted by the Examiner.

The Examiner respectfully traverses this argument as follows.

The Silberschatz reference, as shown above, teaches these limitations as well known in the art. It would have been obvious to a person of ordinary skill in the art to adjust the branch targets as specified in the claim language because failure to do so would render the prior art inoperable.

Applicants' arguments have been fully considered but have been found unpersuasive.

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. § 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

The factual inquiries set forth in *Graham* **v.** *John Deere Co.*, 383 U.S. 1, 148 USPQ 459 (1966), that are applied for establishing a background for determining obviousness under 35 U.S.C. § 103(a) are summarized as follows:

1. Determining the scope and contents of the prior art.
2. Ascertaining the differences between the prior art and the claims at issue.
3. Resolving the level of ordinary skill in the pertinent art.
4. Considering objective evidence present in the application indicating obviousness or nonobviousness.

This application currently names joint inventors. In considering patentability of the claims under 35 U.S.C. § 103(a), the examiner presumes that the subject matter of the various claims was commonly owned at the time any inventions covered therein were made absent any evidence to the contrary. Applicant is advised of the obligation under 37 CFR 1.56 to point out the inventor and invention dates of each claim that was not commonly owned at the time a later invention was made in order for the examiner to consider the applicability of 35 U.S.C. § 103(c) and potential 35 U.S.C. § 102(e), (f) or (g) prior art under 35 U.S.C. § 103(a).

3. Claims 2-24 are rejected under 35 U.S.C. § 103(a) as being unpatentable over US Patent No. 6,370,687 to Shimura (supplied by Applicant) and further in view of Official Notice.

Regarding claim 2, Shimura teaches:

A networked system (column 2, lines 16-30) comprising:

a network (column 2, lines 16-30):

a server coupled to the network, wherein the server includes (column 2, lines 16-30):

an application code source that stores a client application, and a server code manager coupled to the application code source (Web server from which programs are retrieved, column 4, lines 27-35; Fig. 1, reference 20);

an application code transformation manager coupled to the application code source, for transforming the client application from a first format to a native binary format compatible with a native instruction set of the CPU of the client ("compile controller ... recognizes the execute form of the client ... and recognizes the native code of the execution environment in which the compile unit 28 compiles from the byte code of the Java program"; column 4, line 52 – column 5, line 26); and

a server code segment manager coupled to the application code transformation manager, for parsing the client application in the native binary format into a plurality of code segments, said parsing of said code segments being dynamically performed based on actual server-side and client-side execution overhead, network bandwidth efficiency and client-side storage requirements on a per client basis, and configured based on predicted code segment usage or prior code segment usage history, wherein at least one of said plurality of code segments being transmitted to the client via the network [*"Then, in response to a request for the Java program from a client to the Web server on the network, the substitute compile server 10 returns to the client the Java program which has been* ***compiled and optimized into a native code conforming to the execute form of the requester client.***" (column 6, lines 25-48, emphasis added)]; and

a client coupled to the network said client not having said client application stored thereon, wherein the client comprises (column 2, lines 16-30):

a CPU for natively executing at least one of said plurality of said code segments derived from the client application stored on said server (column 4, line 52 – column 5, line 26);

a code cache coupled to the CPU for storing said code segments (Official notice is taken that it is well known in the art to provide a CPU with a code cache. This Official Notice was first taken in the office action of 16 May 2005. Applicants have not traversed this use of Official Notice and as such, it is regarded as admitted prior art. See MPEP 2144.03 (C)); and

a client code manager-coupled to the code cache, for launching the client application by requesting that the server code manager transmit at least one of the plurality of dynamically tailored code segments to the client (column 5, lines 27-32), receiving at least one of the dynamically tailored code segment from the server (column 5, lines 58-61), storing the dynamically tailored code segment in the code cache (Official Notice has been taken above regarding a code cache), and executing at least one of the plurality of dynamically tailored code segments using the CPU until the executed dynamically tailored code segment attempts to pass control to a required code segment not stored in the code cache (column 7, line 42 – column 8, line 18), at which point control passes back to the client code manager to retrieve the required code segment from the server, with the CPU continuing execution with the required code segment [*"...if the client side makes a request for the class file 42-2* [a next required code segment] *with successful prediction, then the compiled Java program can be executed at a high speed in an immediate response to the program request from the client side sine that program has already been compiled and retained on the cache unit 12."* (column 7, line 42 – column 8, line 18)].

It would have been obvious to a person of ordinary skill in the art at the time of Applicants' invention to implement the system taught by Shimura using a code cache in conjunction with the list of processors expressly taught by Shimura [*"SPARCV8, X86, Intel 486, etc."* (column 4, line 52 – column 5, line 26) because these processors either possess a code cache or it is well known in the art to use a code cache. The motivation for using a code cache is well known in the art, for example, to increase the effective speed of the processor by caching instruction code on the processor.

Regarding claim 3, Shimura teaches that the first format is other than the native execution format of the CPU of the client (column 5, lines 15-26). A compiler is functionally equivalent to a "transformation engine to transform the client application from the first format to the native binary format of the CPU of the client".

Regarding claim 4, Shimura does not explicitly teach that the first format is a source code text format of a programming language and the transformation manager comprises a compiler that compiles and links the client application into a native binary format of the CPU of the client. However, Shimura does explicitly teach that the first format is a "Java program in the form of the virtual machine computer program prepared as an applet on the web page" (column 4, lines 28-30) which is compiled using a Java™ compiler (column 4, lines 42-51 and throughout). It would have been obvious to a person of ordinary skill in the art that the term "Java applet" commonly refers to source code in a text format intended for use in a web page and that source code in a text format is well known input to a typical compiler. It therefore would have been obvious to a

person of ordinary skill in the art to implement Shimura's system where the first format is a source code text format of a programming language and compiling that source code into a native binary format of the CPU of the client.

Regarding claim 5, Shimura teaches that the transformation manager comprises a just-in-time compiler (column 5, lines 8-15).

Regarding claim 6, Shimura's teaching regarding class files (column 7, line 42 – column 8, line 18) would be obvious to a person of ordinary skill in the art at the time of Applicants' invention as functionally equivalent to "code segments". It would be obvious to a person of ordinary skill in the art at the time of Applicants' invention to implement this functionality with a client code manager that requests needed segments from the server and to branch into the received code segment. Indeed, this is the functionality implied by Shimura (column 7, line 57 – column 8, line 13) although the obvious details of implementation are omitted.

Regarding claim 7, Shimura does not explicitly recite the steps of adjusting branch instructions to link into and out of received code segments as recited. Shimura implies this functionality (column 7, line 42 – column 8, line 18; column 9, line 63 – column 10, line 9). Official notice is taken that the need to link code that is compiled in sections is well known. It would have been obvious to a person of ordinary skill in the art at the time of Applicants' invention, in combination with his own knowledge of the particular art, to adequately support linking sections of compiled code by adjusting the branch instructions. Failure to do so would

create an inoperable system, as would be recognized as well known by a person of ordinary skill in the art. Applicants have not traversed this use of Official Notice and as such, it is regarded as admitted prior art. See MPEP 2144.03 (C).

Claim 8 recites what is generally known in the art as "garbage collection". Official notice is taken that Java™ and the Java™ virtual machine support garbage collection. It would have been obvious to a person of ordinary skill in the art at the time of Applicants' invention, in combination with his own knowledge of the particular art, to implement the system taught by Shimura using garbage collection because of the well-known advantages of garbage collection, such as ease of programming and recovery unused memory.

Claims 9-13 recite the server portion of the system of claims 1-5 and are rejected for the same reasons given above for claims 1-5.

Claims 14-17 recite the client portion of the system of claims 1 and 6-8 and are rejected for the same reasons given above for claims 1 and 6-8.

Claims 18-22 recite the methods performed by the system of claims 1-7 and are rejected for the same reasons given above for claims 1-7.

Claims 23-24 recite a computer program product and system according to claims 1-7 and are rejected for the same reasons given above for claim 1.

Conclusion

THIS ACTION IS MADE FINAL. Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Jason Proctor whose telephone number is (571) 272-3713. The examiner can normally be reached on 8:30 am-4:30 pm M-F.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Paul Rodriguez can be reached at (571) 272-3753. The fax phone number for the organization where this application or proceeding is assigned is (571) 273-8300.

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100. Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR)

system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Jason Proctor
Examiner
Art Unit 2123

jsp

PAUL RODRIGUEZ
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100